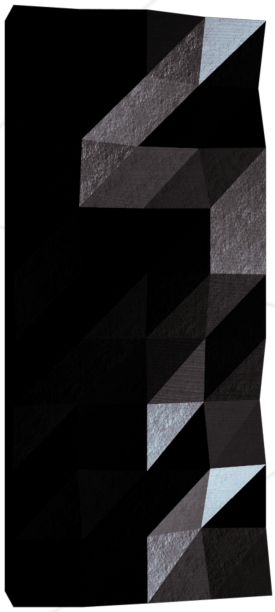

Module 01 – Piscine Python for Data Science

Intro to Python: Syntax and Semantics

Summary: This day will help you to get the basic knowledge about syntax and semantics of Python.

Contents

Chapter I

Foreword

Python is the most popular programming language for data science. Why is it so good for that kind of task? Python is an interpreted language. It means that you can easily interact with different pieces of code and get fast results. And that is exactly what we need if we want to analyze data from different angles or try different hyperparameters for a machine learning model. Besides this, Python has a lot of libraries that are suitable for scientific tasks including data science. Add to this a pretty simple syntax and you will get the most popular programming language for data science tasks.

Just for fun, look at these 19 beautiful guiding principles that influenced the design of Python:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one– and preferably only one –obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.

- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea – let's do more of those!

In case you forget any of them, you can just write in Python import this and you will get them shortly.

Chapter II

Instructions

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.
- Here and further we use Python 3 as the only correct version of Python.
- The python files for python exercises (module01, module02, module03) must have a block in the end: if ___name___ == '___main___'.
- Pay attention to the permissions of your files and directories.
- To be assessed your solution must be in your GIT repository.
- Your solutions will be evaluated by your piscine mates.
- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your .gitignore to avoid accidents.
- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.
- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google.
- You can ask questions in Slack.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- And may the Force be with you!

Chapter III

Specific instructions of the day

- No code in the global scope. Use functions!
- Each file must be ended by a function call in a condition similar to:

```
if __name__ == \'__main__\':
    your_function( whatever, parameter, is, required )
```

- It is tolerated to place an error handling in this same condition.
- No imports allowed, except those explicitly mentioned in the section 'Authorized functions' of the title block of each exercise.
- You can use any built-in function if it is not prohibited in an exercise.
- The exceptions raised by the open() function are not to be handled.
- The interpreter to use is Python 3.

Chapter IV

Exercise 00 : Data types

	Exercise 00
Data types	
Turn-in directory : $ex00/$	
Files to turn in : `data_types.py`	
Allowed functions : `n/a`	

- Python as any other language has several built-in data types. In this exercise, you will get familiar with the most popular and useful ones.
- Create a script called data_types.py in which you need to define a data_type() function. In this function, you need to declare 8 variables with different types and print their types on the standard output.
- You must reproduce exactly the following output:

```
> python3 data_types.py
[int, str, float, bool, list, dict, tuple, set]
```

- It is forbidden to explicitly write the data types in your print. Remember to call your function at the end of your script as explained in the day instructions:

```
if __name__ == '__main__':
    data_types()
```

- Put your file in the ex00 folder in the root of your repository.

Chapter V

Exercise 01 : Working with files

	Exercise 01
Working with files	
Turn-in directory : $ex01/$	
Files to turn in : `read_and_write.py`	
Allowed functions : `n/a`	

- For this exercise, you are free to define as many functions as you need and to name them whatever you want. In the attached file ds.csv (you will recognize it from the previous day experience), you will have several columns separated by a comma with different data about vacancies.
- Design a Python script called read_and_write.py whose role is to open the file ds.csv, read the data it contains, replace all the comma delimiters with ''^and save it to another file ds.tsv. Be careful, your data may contain commas. If you replace them, you will corrupt the data.
- Put your script in the ex01 folder in the root of your repository.

Chapter VI

Exercise 02 : Search by key

	Exercise 02
Search by key	
Turn-in directory : $ex02/$	
Files to turn in : `stock_prices.py`	
Allowed functions : `import sys`	

- You have the following dictionaries to copy to one of your functions:

```
companies = {
    'Apple' : 'AAPL',
    'Microsoft' : 'MSFT',
    'Netflix' : 'NFLX',
    'Tesla' : 'TSLA',
    'Nokia' : 'NOK'
 }

stocks = {
        'AAPL' : 287.73,
        'MSFT' : 173.79,
        'NFLX' : 416.90,
        'TSLA' : 724.88,
        'NOK' : 3.37
}
```

- Write a program that takes as argument a name of a company (ex: Apple) and displays on the standard output the stock price (ex: 287.73). If you give the program as an argument a company that is not from the dictionary, your script should display Unknown company. If there are no arguments or too many arguments, your program should do nothing and quit.

```
$> python3 stock\_prices.py tesla
724.88
$> python3 stock\_prices.py Facebook
Unknown company
$> python3 stock\_prices.py Tesla Apple
```

Chapter VII

Exercise 03 : Search by value and by key

	Exercise 03
Search by value and by key	
Turn-in directory : $ex03/$	
Files to turn in : `ticker_symbols.py`	
Allowed functions : `import sys`	

- You have the same two dictionaries from the previous exercise. You should copy them again in one of your functions of the script.
- Create a program this time that takes a ticker symbol (ex: AAPL) and displays the company name and the stock price with space as the delimiter. The rest of the behavior must be identical to the previous exercise.

```
$> python3 ticker\_symbols.py tsla
Tesla 724.88
$> python3 ticker\_sybmbols.py FB
Unknown ticker
$> python3 ticker\_symbols.py TSLA AAPL
```

Chapter VIII

Exercise 04 : Dictionaries

	Exercise 04
Dictionaries	
Turn-in directory : $ex04/$	
Files to turn in : `to_dictionary.py`	
Allowed functions : `n/a`	

- Create a script named to_dictionary.py where you need to copy in one of your functions the list of the following tuples as is:

```
list_of_tuples = [
            ('Russia', '25'),
            ('France', '132'),
            ('Germany', '132'),
            ('Spain', '178'),
            ('Italy', '162'),
            ('Portugal', '17'),
            ('Finland', '3'),
            ('Hungary', '2'),
            ('The Netherlands', '28'),
            ('The USA', '610'),
            ('The United Kingdom', '95'),
            ('China', '83'),
            ('Iran', '76'),
            ('Turkey', '65'),
            ('Belgium', '34'),
            ('Canada', '28'),
            ('Switzerland', '26'),
            ('Brazil', '25'),
            ('Austria', '14'),
            ('Israel', '12')
]
```

- Your script should transform this variable into a dictionary where the number is the key and a country is the value. The same key may have several values as you see. The script must display the content of the dictionary on standard output accordingly to this precise formatting:

```
'25' : 'Russia'
'132' : 'France'
'132' : 'Germany'
'178' : 'Spain'
...
```

- Think about why the order is not necessarily identical to the example.

Chapter IX

Exercise 05 : Search by value or by key

	Exercise 05
Search by value or by key	
Turn-in directory : $ex05/$	
Files to turn in : `all_stock.py`	
Allowed functions : `import sys`	

- You still have those two dictionaries from ex02. And you still should copy them in one of your functions in the script.
- Write a program that has the following behavior:
  - the program must take as argument a string containing many expressions to find whatever you want, separated by a comma
  - for each expression of the string, the program must detect if it is a company name or a ticker symbol, or neither
  - the program should not be case-sensitive and be able to work with white spaces
  - if there are no arguments or too many arguments, the program displays nothing
  - when there are two commas in a row in the string, the program does not display anything
  - the program must display the results separated by a line break and use the following formatting:

```
$ python3 all\_stocks.py 'TSLA , aPPle, Facebook'
TSLA is a ticker symbol for Tesla
Apple stock price is 287.73
Facebook is an unknown company or an unknown ticker symbol
$ python3 all\_stocks.py 'TSLA,, apple'

$ python3 all\_stocks.py 'TSLA, , apple'

$ python3 all\_stocks.py TSLA AAPL
```

Chapter X

Exercise 06 : Sorting a dictionary

	Exercise 06
Sorting a dictionary	
Turn-in directory : $ex06/$	
Files to turn in : `dict_sorter.py`	
Allowed functions : `n/a`	

- In this exercise, you need to take the list of tuples from the ex04 with the countries and numbers and make a dictionary out of it where the countries are keys and the numbers are values . You should copy it in one of your functions in the script.
- Write a program that displays the country names sorted by descending numbers, then in alphabetical order of names if the numbers are equal. You need to display them one per line and without the numbers:

```
The USA
Spain
Italy
France
Germany
...
```

Chapter XI

Exercise 07 : Sets

	Exercise 07
Sets	
Turn-in directory : $ex07/$	
Files to turn in : `marketing.py`	
Allowed functions : `import sys`	

In this exercise, imagine that you work in a marketing department. You will operate with different lists of email accounts. The first list is your clients email accounts. The second list contains the email accounts of your last event participants (some of them were your clients). The third list contains the accounts of your clients that viewed your last promotional email.

In business terms, you need to:

- Create a list of those who have not seen your promotional email yet. The list will be sent to the call center to reach those people.
- Create a list of the participants who are not your clients. You will send them an introduction email about your products.
- Create a list of the clients who did not participate in the event. You will send them the link to the video from the event and the slides.

  Technical details:

- Create different functions that convert your lists to sets and use set operators that you need to use to perform the aforementioned business tasks and return the required lists of email accounts.
- Arrange your code in a script. The script takes as an argument the name of the task to perform: call_center, potential_clients, loyalty_program. If the wrong name is given, raise an exception.
- For the exercise you need to use the following three lists:

```
clients = ['andrew@gmail.com', 'jessica@gmail.com', 'ted@mosby.com', 'john@snow.is', 'bill\
    _gates@live.com', 'mark@facebook.com', 'elon@paypal.com', 'jessica@gmail.com']
participants = ['walter@heisenberg.com', 'vasily@mail.ru', 'pinkman@yo.org', 'jessica@gmail.com', '
    elon@paypal.com', 'pinkman@yo.org', 'mr@robot.gov', 'eleven@yahoo.com']
recipients = ['andrew@gmail.com', 'jessica@gmail.com', 'john@snow.is']
```

Chapter XII

Exercise 08 : Working with strings as lists

	Exercise 08
Working with strings as lists	
Turn-in directory : $ex08/$	
Files to turn in : `names_extractor.py, letter_starter.py`	
Allowed functions : `import sys`	

- Imagine that you work in a corporation where the email accounts always have the same template: name.surname@corp.com.
- Create a script that takes the path to a file with such email accounts as an argument. All the emails are delimited by '\n' in the file. The script should return a table with the fields: Name, Surname, E-mail delimited by '\t'. Names and surnames values should start from a capital letter. The table should be stored in the file employees.tsv.
- The example:

```
Name    Surname    E-mail
Ivan    Petrov    ivan.petrov@corp.com
Emma    Geller    emma.geller@corp.com
John    Smith    john.smith@corp.com
```

- Create another script that takes in an email, searches the corresponding name from the file created by the first script and returns the first paragraph of a letter:
- *Dear Ivan, welcome to our team. We are sure that it will be a pleasure to work with you. Our company hires only that kind of professionals.*
- It is prohibited to use the structure 'la-la 0'.format(text). Please, use f-strings. They are more readable and faster.

Chapter XIII

Exercise 09 : Caesar cipher

	Exercise 09
Caesar cipher	
Turn-in directory : *ex*09/	
Files to turn in : `caesar.py`	
Allowed functions : `import sys`	

- There is such a thing as Caesar cipher that helps encode some text using a shift in the alphabet order. For example, the encoded version of hello might be tqxxa if we use the shift equal 12.
- Write a program that will encode any string using a given shift or will decode any string using a given shift accordingly to the argument given:

```
$ python3 caesar.py encode 'ssh -i private.key user@school21.ru' 12
   eet -u bduhmfq.wqk geqd@eotaax21.dg
$ python3 caesar.py decode 'eet -u bduhmfq.wqk geqd@eotaax21.dg' 12
   ssh -i private.key user@school21.ru
```

- If the scripts are given a string with, for example, Cyrillic symbols, the scripts should raise the exception The script does not support your language yet. If an incorrect number of arguments is given, raise an exception